MICROLOGIC FAMILIARIZATION FOR SALES PERSONNEL

I. Preface

A. Purpose

The basic purpose of this micrologic familiarization program is to help you sell micrologic. We hope through this program to give you a very general understanding of digital computers, and the terminology which goes with them. We do not plan to make experts of you in the next couple of hours. We do hope to make you familiar with micrologic - what it is, how it fits in to the scheme of things in computer construction, and why micrologic is a useful approach to computer construction. You, as salesmen, will be our primary contact with those who might use our micrologic elements. In the sale of transistors and diodes, a rudimentary knowledge of the devices as well as their use in circuits is necessary. Similarly, in the case of micrologic elements, a rudimentary knowledge of these devices and their use in computer systems, is necessary. We plan to have a fairly extensive applications backup for you in the sale of micrologic elements. The nature of this backup and the information that is available now will be described later.

B. The Customer

Before getting into a discussion of micrologic, it might be worthwhile spending some time in a description of what you can expect to encouter in the customer. Many of you, I am sure, have already run into some or all of the points that I will discuss below. To start with, he's a very busy man. That is, if you've got the right guy. He may be a logical designer, a systems designer - an advanced planning man, he may even be a circuit designer; but he is busy. Secondly, he has been and continues to be besieged with propaganda on behalf of the various microminiaturization approaches. He is even beginning to receive some under-the-table propaganda against other approaches. He often has some of his own company's or some government money to look into microminiaturization through feasibility studies. With all the slick paper staring him in the face, he has probably lost sight of which way is up, and wishes that he

had never heard of microminiaturization. He will, however, look at almost anything that seems promising if the price is right. He will look, that is, one time; if it doesn't work that one time, he will be a long time coming back for a second look. He is becoming confused as to what to look for. He has been told by the advocates of every approach that no other approach will ever be cheap, that no other approach will be reliable, that no other approach will be anywhere near as useful. He has even been told that he doesn't need the reliability because what the approach specified is so small he can build in redundancy. All he really wants to do is find a simple method of building a reliable microminiature system. This is where you come in to the picture. If you can get your foot in the door, then we can come in to the picture.

C. Topics to be Discussed

1. Introduction to Computers
2. Micrologic
3. Applications Support
4. Comparison of Micrologic to other Approaches

II. Introduction to Computers

A. What are Computers

Simply stated, computers are machines for performing calculations. A digital computer is a machine which calculates with numbers. The word digital comes from digits which carries two implications; first digits meaning meaning fingers, that is, the fingers that you count on when you make calculations, and secondly the step by step operation of a digital computer.

There are basically four varieties of digital machines. Some machines can perform a wide variety of calculations. Furthermore, operators of these machines have the widest possible control over these calculations. These are called general-purpose computers.

The second set, which is really a sub-set of the first, include machines which are optimized for special applications such as accounting or scientific or what have you. The third set includes all machines which are designed to perform one set, or one particular group of sets, of calculations. These are known as special purpose computers. The types of calculations which they can perform are often as varied as those of the general purpose computer; but the operator has little or no control over the operations which are performed. The fourth set of digital machines is not properly called computers. It includes all machines which perform some special data processing operation, for example, digital smoothers, digital radar detectors, and even includes such things as voltmeters, counters and desk calculators.

B. How a Computer Operates

The operation of a digital computer will be explained by following the step-by-step method of solution to a simple problem.

Let us suppose that you have the equation

$$Y = X^3 - 3X + 2$$

The problem would be, in the case taken here, to solve for the roots of this equation or those values of X for which Y = 0. There are many methods of solving this equation which you would use, and also, which a computer would use. However, the explanation is simplified if we take the Brute Force Method, that is, we will substitute various values of X in this equation until Y = 0, which is a method that you might use. Let us start by substituting X = 0 in calculating Y, and then incrementing X by 0.2 calculating Y, incrementing again by 0.2, etc. Now you might approach the problem this way, with the scratching shown below:

$x = 0$ $\quad$ $x = .2$

$y = 2$ $\quad$ $y = .008$

$-.120$

$+2.04$

1.888

$.04$

Most likely, to better organize your work your calculations would appear as shown below:

$$Y = X^3 - 3X + 2$$

A	B	C	D	E	F	G
X	X^2	X^3	-3X	X^3 -3X (D+C)	X^3 -3X + 2 (E+β)	Y
0	0	0	0	0	2	2
+.2	.04	.008	-.12	-.112	1.888	1.888
+.4	.16	.064	-.48	-.416	1.584	1.584
...	...	....	....	.....	.....	.....

The use of a table such as this is by far the preferred way of organizing the solution to a problem. Each step in the table shown above is simple arithmetic. To solve the problem above, you would keep incrementing X until Y goes to zero or goes negative. If it went negative, you would know that the root was somewhere between the last value X and the one for which it went negative, and you could interpolate between or calculate with smaller increments between. Of course in this problem, actually, Y goes to zero.

If you had a desk calculator at your elbow, you could save some work by having it do the arithmetic for you. A calculator, of course, would just replace your fingers and scratch pad in performing each calculation; but the operations remain the same.

If you had on hand a digital computer, you could perform the calculation much faster. Again, the computer would just be replacing your fingers and scratch pad, performing the same operations. With the computer, though, you have reached the point of diminishing returns. It calculates so fast that you're wasting time.

The computer could cost as much as a dollar a minute to operate. It takes you minutes to tell the computer what to do next, and it takes a computer millionths of a second to perform the operation. So, instead of telling a computer one step at a time, what operations are to be performed, you tell it at one time all the operations that are to be performed; and then let the computer recall the operations that you've told it and perform them sequentially, to come to a solution. You must be careful, of course, to list all the operations you actually perform. This is something like balancing a checkbook, as one goes along calculating their checkbook balance making additions and subtractions, we usually find that we can perform some of the additons and subtractions (mostly subtractions) in our head; but if we did that, the desk calculator could'nt keep track of the balance correctly, so all of them must go in on the desk calculator. Suppose we list the operations that we actually perform in solving this equation:

1. Start

   Set X = 0 in Column A

2. Compute

   (a) Take contents of Column A
   (b) Multiply by contents of Column A
   (c) Put product in Column B
   (d) Multiply product by contents of Column A
   (e) Put product in Column C
   (f) Take contents of Column A
   (g) Multiply by -3
   (h) Put product in Column D
   (i) Add the contents of Column C to the product
   (j) Add two to the resulting sum
   (k) Put this sum in Column G

This, of course, gives us Y. Now we are ready to make a decision.

3. Decision

(a) Look at Y, and if positive, go to (b); if negative, go to (f).

(b) Take contents of Column A

(c) Add 0.2

(d) Put the sum in Column A

(e) Go back to compute (2)

(f) At this point, we can stop or we can go back using smaller increments in X to get a better resolution on X. In either case, we print out the resulting X, or take it to use in other calculations.

Remember that all this above is just a literal description of the operations we actually perform in finding the roots of the equation given. This is a program.

Since the computer can't read our table, which acts as a storage for us, it contains within it an electrical table or memory which is used the same way. We might, in fact, think of the letters which we use as columm designations in our table as what are known as memory addresses in the storage of the computer. In addition, since we've already said that we must tell the computer all the operations it will perform instead of proceeding one operation at a time, we must also store this list of operations or program; and they will be stored in other addresses in the memory.

The program we store might look something like this:

Program Address	Operation	Instruction Code
a	Take contents of t (+0)	(ct)
b	Transfer to A	(ta)
c	Take contents of A	(cA)
d	Multiply by contents of A	(mA)
e	Put in B	(tB)
f	Multiply by contents of A	(mA)
g	Put in C	(tc)
h	Take contents of A	(cA)
i	Multiply by contents of u (-3)	(mu)
j	Add the contents of C	(aC)
k	Add the contents of v (+2)	(av)
l	Store result in G	(tG)
m	Transfer to address r if Y is negative, otherwise go on to n	(yr)
n	Take contents of A	(cA)
o	Add the contents of w (+0.2)	(aw)
p	Put in A	(tA)
q	Go back to order at address c	(xc)
r	Print the contents of A	(pA)
s	Stop	(z)
t	+0	
u	-3	
v	+2	
w	+0.2	

We have here then our same old list of operations with perhaps minor changes. Assigned to each operation we have as an address the lower case letters in the left hand column. When this program is inserted into the computer each operation is stored at the address shown. When we wish the computer to perform this operation, we merely say to the computer, "Go to address 'a' and do whatever it says." The computer takes over from there. Instead of writing out these operations in the computer's memory, we use symbols. The instruction codes which appear in the parentheses in the right hand column would represent what is actually stored in the computer memory. Here we have the first letter, always lower case, which tells what the nature of the operation is, multiply or add or what have you. We then have the second letter which tells where in the memory you go to get the number that you will operate on or with. This instruction code then tells the computer exactly what you say in the list of operations. In normal computers we do not use the literal addresses such as those shown here, (as you can see we have almost run out of the alphabet in the program addresses) instead we use numerical addresses.

If we stored the program above for the case

$$aX^2 + bX^2 + cX + D = Y$$

Increment X = K

we would then have a general program capable of solving any third order equation. We would take this general program and insert for our case a = 1, b = 0, c = -3, K = +.2 and have the same result that we had before. This general-type program is what is called a subroutine.

If we did not have to insert this program into the memory but rather have it permanently wired in, we would have what is called wired program or special purpose computer.

We have seen how a digital computer operates to solve problems using simply arithmetic and decision processes. All that a computer can do is summarized in the steps outlined above. They seem like small steps but nevertheless can represent a great deal of work because of the speed at which a computer can perform these steps and the accuracy with which they are performed. To give you some idea of what can be expected normally from a computer, it would take a small fraction of a second to solve the problem above to an accuracy of six decimal digits. We will now take a look at how a computer performs these operations.

C. Organization of a Computer

A digital computer contains three basic types of functional units. The first is a memory of one or more forms which is our electrical table discussed earlier. We then must also have an add-subtract unit which is, as the name implies, capable of adding or subtracting two numbers. We further require gates to route numbers around in the computer so that they arrive at the proper place at the proper time to perform operations. These basic parts of the computer will be discussed in somewhat more detail below.

1. Memory

There are two basic units of size of memory. One is a cell which is the fundamental unit as capable of storing one digit. The next higher order unit of size is the register. This is a group of cells and consequently can store a group of digits. The register length, which is a way of describing the number of digits it will store, usually corresponds to what is known as the word length of the computer. The word length, of course, determines the basic accuracy which can be achieved with the computer, thus a six decimal digit word length would be capable of a one part in a million resolution in the numbers that it works with and in a sense in the solutions that can be achieved.

There are three basic types of memory used in a digital computer. Two are more or less to be considered to be in the permanent category. The first of these is a cyclic memory. The second is known as random access memory. An example of the first permanent-type would be the magnetic drum. Here numbers are stored sequentially on the drum and one must wait until the number comes around before being able to extract it from storage. This is usually the length of time it takes the drum to revolve. This is often satisfactory in most calculations.

At times the time required to get a number from storage is too long and results in slowing the computer to a point which seriously affects its performance. In cases of this type the random access memory, a good example of which is the core memory, is used. Here the number which is stored is available immediately regardless of where in the memory it is stored. The choice between the core and drum memory in a computer is based on the cost vs. performance of each in the solution of the problem. The drum memory is much cheaper than the core memory for every number stored.

The third general type of memory is the temporary memory. A one cell temporary memory would be a flip-flop, for example. A temporary register usually takes theform of a shift register. Shift registers are really just groups of flip-flops having the ability to transfer a number from one flip-flop to the next. They will be covered further below.

2. Add-Subtract Unit

In general, all of the operations on numbers are performed in the add-subtract unit. This is a functional unit which is capable of adding two numbers together or subtracting numbers and having at its output either the sum or difference of the numbers. The other operations such as multiplication and division are merely multiple additions or subtractions.

3. Gates

All numbers pass through gates in the couse of an operation. The gates are in a sense like the switches in a railroad yard. They route numbers or rather determine the route numbers can take in proceeding from one point to another. Two types of gates are generally encountered in computers, AND gates and OR gates. OR gates are used to allow numbers from several points to go through one point, such as a register or add-subtract unit. They are designed so any number that appears at any input to an OR gate will appear at the output. They operate in a sense like summing amplifiers. AND gates, on the other hand, have basically just two inputs, a number and a control. The control input determines whether the number appearing at the other input can go through the AND gate or not.

The operation of these various functional units in a calculation will be illustrated by returning to our original problem and examining in detail what happens inside the computer in its performance of part of the problem. The four figures below, figures 1, 2, 3 and 4 are used to illustrate the step-by-step procedure of starting the computer on the problem and performing the first operation. To start the computer, we must manually insert a start code and also an address which tells the computer where it is supposed to start. Looking at figure 1, we see that the start code has gone into the order code register, and the address at which we want to start the operation, which is a, has gone into the next instruction address register. The start code is decoded in the matrix

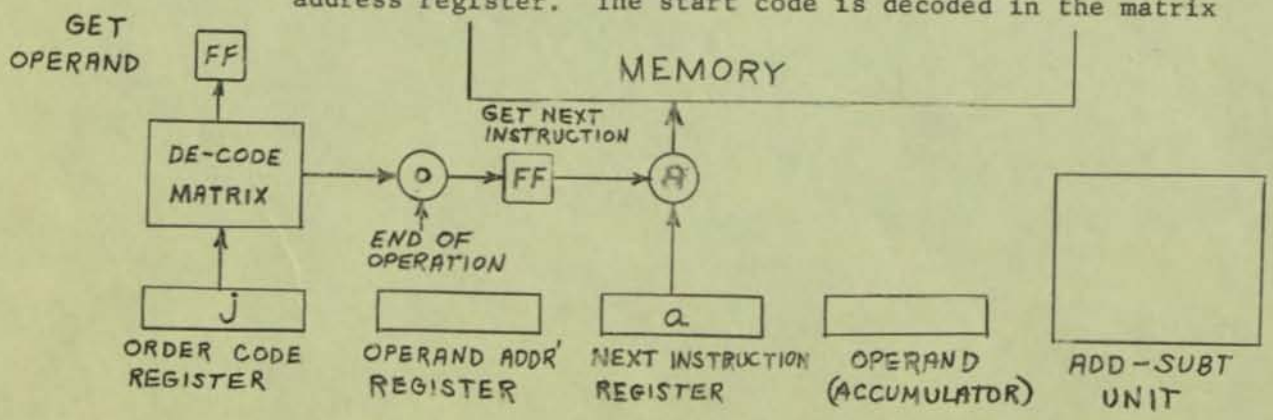


Figure 1. Next Instruction Address to Memory

and sets the flip-flop which says in effect, "Get the next instruction." When this flip-flop is set, the next instruction address is sent into the memory. The memory sends out whatever set of numbers apppears at the memory address that goes in. Since we know that this set of numbers is really the next instruction, we route this number through a gate, (see figure 2) into the order code and operand address registers. The first

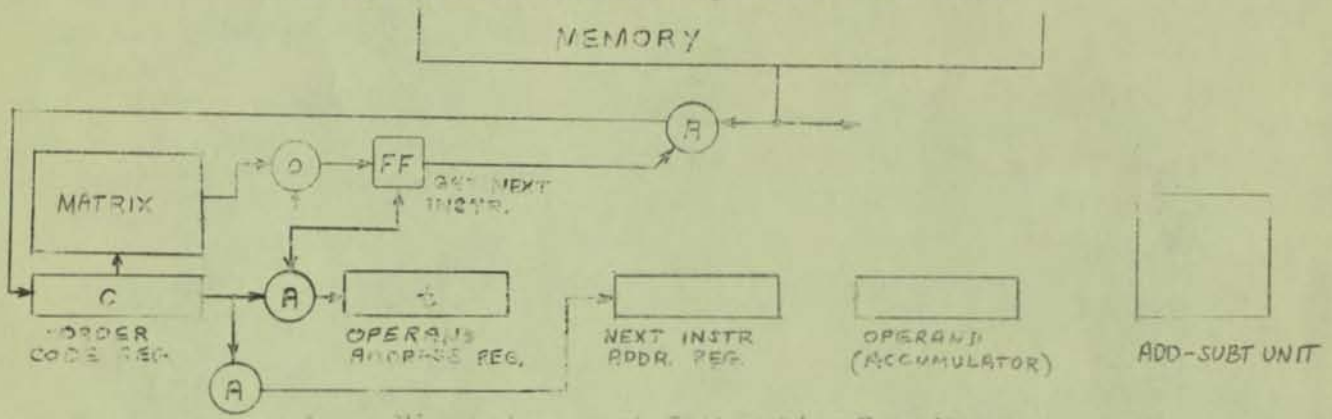


Figure 2. Get Instruction From Memory

instruction code, referring back to the program, was (ct), where c was an order code which meant "get from memory," and t was the position in the memory or the operand address of the number we were supposed to get from the memory. The c order sets a flip-flop which in effect says "get operand." See figure 3.

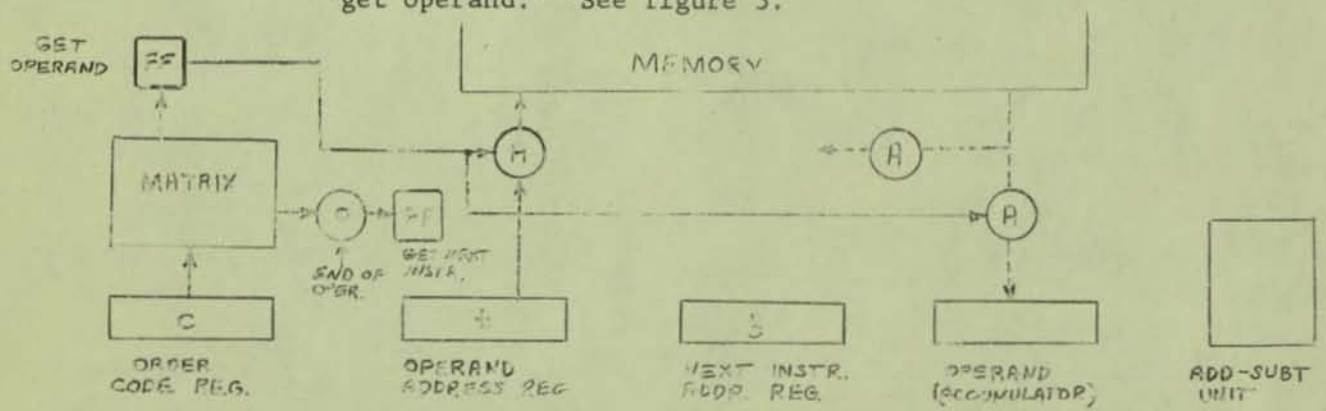


Figure 3. Operand Address to Memory

This flip-flop then opens an AND gate which allows the operand address to go to the memory. This flip-flop also opens the gates which connects the memory with the operand

register or accumulator, thus this time the number which appears at the address that went into the memory goes directly to the operand register. See figure 4.

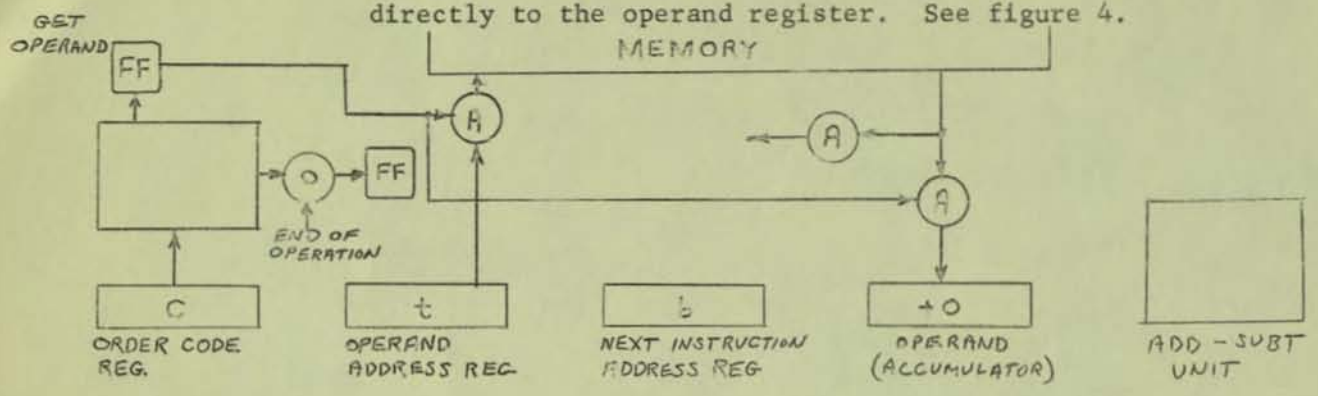


Figure 4. Operand From Memory

While this operation is taking place, the next instruction address, which is in the next instruction address register, has been incremented one so that it is now b. Once the operand is in the accumulator, we get an end of operation signal in through an OR gate into the "get next instruction" flip-flop. We are now ready to repeat the basic cycle and keep repeating it until we conclude the calculation.

The above then demonstrates the use of various types of functional units in a computer. The add-subtract unit as such was not demonstrated. When the order code calls on the computer to add, one of the numbers to be added is in the accumulator, the other is brought out from the memory. The two numbers are passed together through the add-subtract unit and the sum or difference is returned to the accumulator. Since I don't want you taking my job, I won't go any further on computer design.

D. Binary Notation

Up to now we have been dealing solely with decimal numbers. Some computers in fact do calculate decimally. There is no other number form. Most computers use what is called a binary number system. Most computers are constructed with devices which have two stable states like a toggle switch. They may be on or off. The binary number system recognizes and can work with this two state system.

If we look at a decimal number

1 3 4 7

we find that what this number actually represents, in fact sometimes we say it this way, is expressed by the following equation:

$$1 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$

Thus, as we know the number that we wrote was really a hand writing expressing the above equation which represents a quantity, we say that the decimal system has a modulus of ten, that is that the numbers that we write are the coefficient of powers of ten. (We don't write, for example, 11×10^2, since this is really 1×10^2 plus 1×10^2.) In the binary number system the numbers we write are coefficients of powers of 2, thus a number

0 1 0 1 1

is really the hand writing of the expression

$$0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Of course, we can calculate readily what this number is in decimal form.

$$0 + 8 + 0 + 2 + 1 = 11$$

Once again, as in the case of the decimal system, we can't say 2×2^2 because 2×2^2 is really 1×2^3, thus the binary coefficients can only have two values, 0 and 1 as in the case before. This is eminently suited to devices such as on-off switches on diode gates and so forth which are most easily designed to have two states. Arithmetic operations with binary numbers are identical to those with decimal numbers. We will use some examples of addition, subtraction and multiplication to demonstrate the identity of the method. The first example will be in addition

$$\begin{array}{r} 10010 \\ +01011 \\ \hline \end{array}$$

It will help you to understand binary notation, if you work this problem out yourself. Use the same rules that you do with decimal arithmetic. If you have done this arithmetic properly, the solution you get will be

11101

The next problem will be one in substraction:

```
 11011
-10110
```

If you have done this subtraction properly, keeping in mind that the methods are identical to those of decimal subtraction, then the answer you will get will be:

00101

Multiplication with binary numbers is much simpler than it is with decimal numbers. Again, the rules are identical; but here, we are only working with the numbers 0 and 1, and multiplication by either number is very simple. To demonstrate:

```
  10010
 x01100
 -----
  00000
 00000
10010
10010
00000
-------
011011000
```

The method of multiplication should be fairly obvious from the numerical illustration given above.

E. Computer Logic and How Micrologic Fits In

The only portion of a computer which usually does not contain logic circuitry is the permanent memory. Some logic circuitry is used in getting in and out of the memory; but the memory itself, in general, uses no logic circuitry. Except for the memory as shown above, pretty near all computer circuitry is logic circuitry of some form. The

functional units, which were mentioned earlier, which do contain logic circuitry, will now be described in somewhat more detail. In addition, the manner in which micrologic is used to perform the same function will be illustrated.

1. Shift Registers

A shift register, which is a temporary storage unit, as mentioned earlier, is really a set of flip-flops with a means of transferring information from one flip-flop to the next. The process of shifting can best be described using figure 5 below:

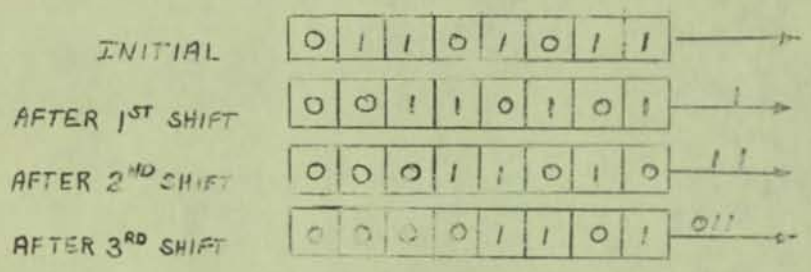


Figure 5. Shift Register Operation

The boxes shown in the shift register are the individual flip-flops in the register. You will observe that initially the binary number is stored in this register in the forms of the one and zero states of the individual flip-flops. After the first shift, the state of each flip-flop has been transferred to the flip-flop on its right. The state of the flip-flop at the right end has gone out on the output line. After the second shift pulse, this process has been repeated, etc. You can see that in this way we can transfer, one binary digit at a time, a number out of a shift register into some other portion of a computer. Similarly, we can transfer a number into a shift register starting at the left end. One common shift register

configuration is shown in figure 6:

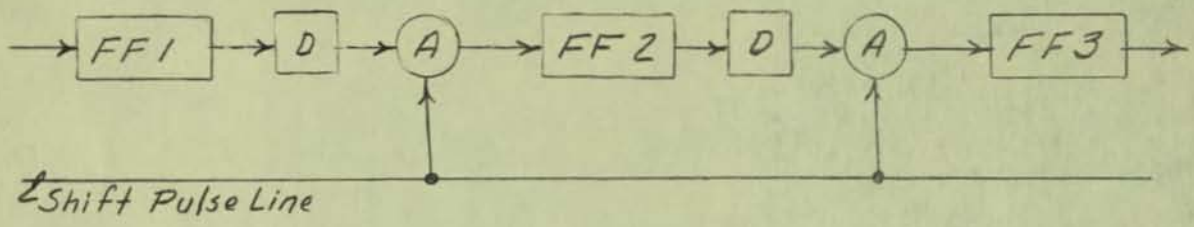


Figure 6. Typical Gated Shift Register Configuration

As shown here, the state of flip-flop 1 is passed through a delay line to one input of an AND gate. When the shift pulse comes along a shift pulse line, the AND gate is turned on and that state is transferred into flip-flop 2. Similarly, the state of flip-flop 2 is transferred into flip-flop 3. The purpose of the delay line is to maintain at the input to the AND gate the previous state of the flip-flop, even though it may have changed when a new state was transferred into it. A second typical shift register configuration, the master-slave arrangement, is shown in figure 7:

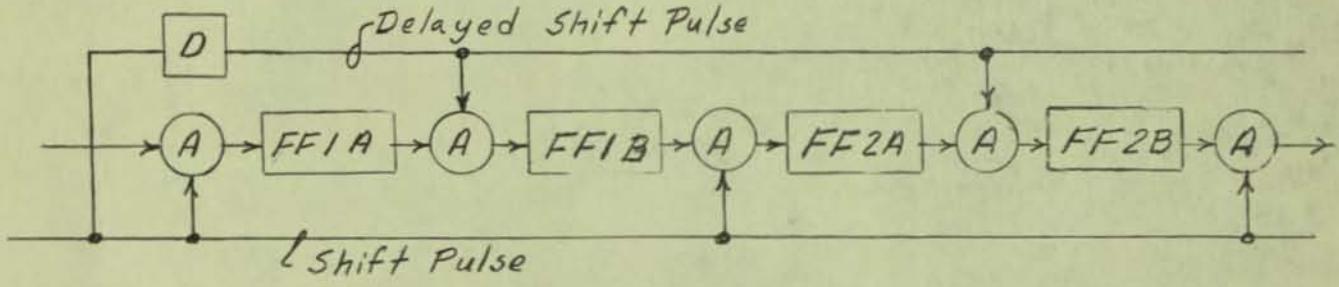


Figure 7. Typical Master-Slave Shift Register Configuration

Here flip-flops are used in place of delay lines at every state. The state of each A flip-flop, which corresponds to the flip-flops in figure 6, is

transferred to the B flip-flops in between shift pulses using the delayed shift pulse. These B flip-flops then serve as a memory during the main shift pulse shifting operation. A typical gated shift register stage implemented using transistor resistor logic is shown in figure 8:

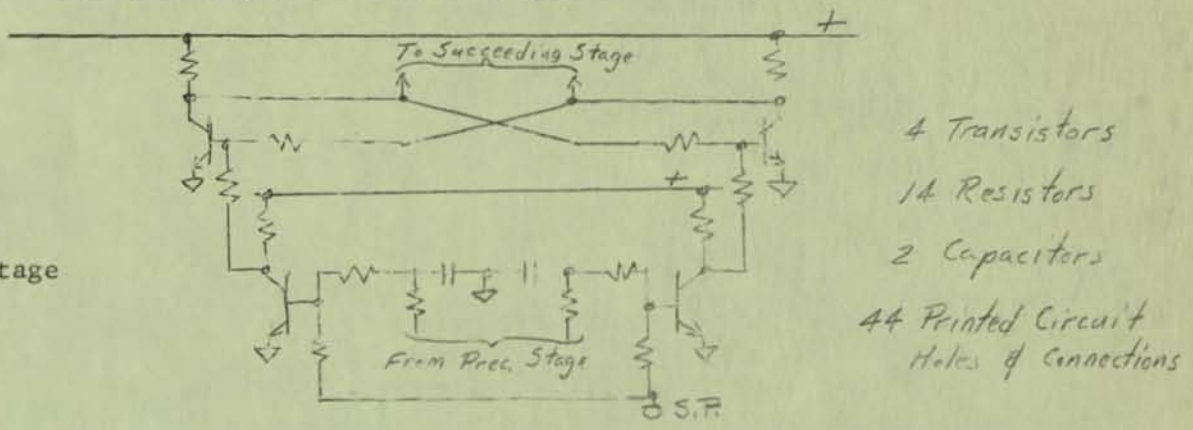


Figure 8.

TRL Shift Register Stage

The RC networks shown in this diagram are the delay lines of figure 6. These RC networks are usually sufficient as a delay in logic circuitry. The additional circuitry required to form the master-slave type shift register in TRL would be extensive compared to the circuitry shown here. A typical shift register stage using micrologic is shown in figure 9:

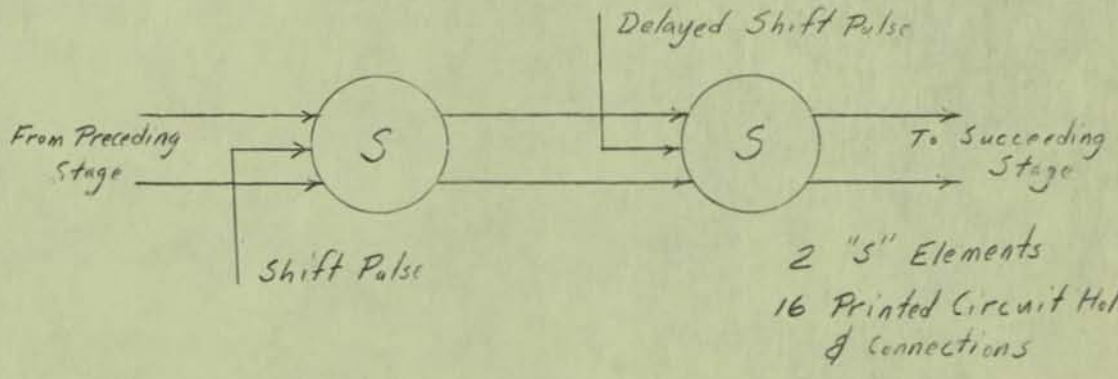


Figure 9. Micrologic Shift Register Stage

Here it is evident that the master slave technique is used. Delay lines compatible with use in this logic circuitry would not only be much larger, but at least as expensive as a micrologic element. Consequently, the master-slave technique is preferred. A variation of this technique using the built-in propagation delay in the micrologic element is shown in Figure 10.

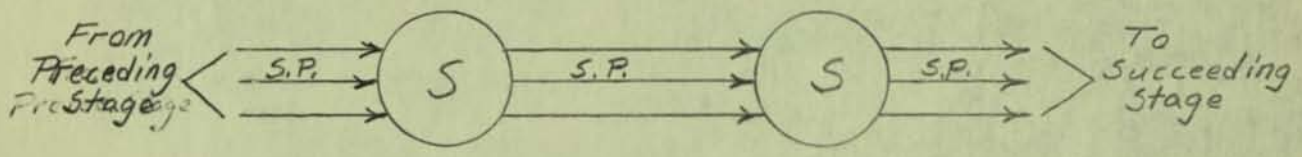


Figure 10. Variation of Micrologic Shift Register Stage

2. Gates

As mentioned earlier, gates are logic circuits used to set up and define the routes that numbers may take going from one portion of a computer to another. The action of OR gates can be explained using Figure 11:

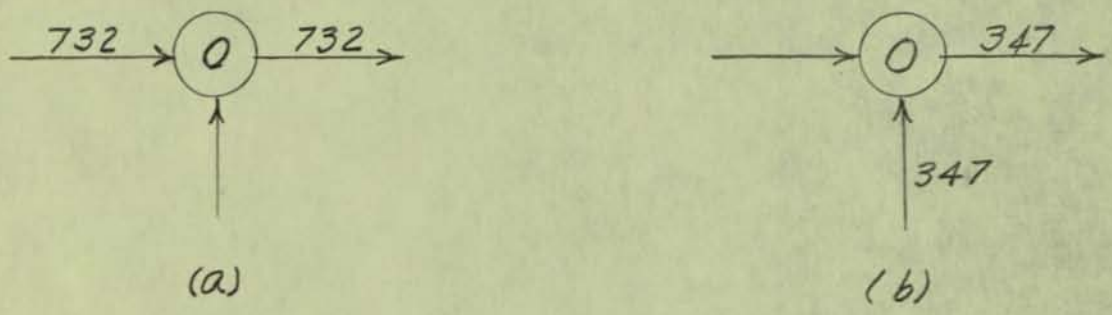


Figure 11. OR Gates

When a set of numbers, in this case 732, appear at one input to the OR gate, the same numbers 732 appear at the output on the right hand side as shown in Figure 11.

In Figure 11b, when the set of numbers 347 appear at the other input to the OR gate, this set of numbers 347 appear at the output on the right hand side. Thus, any number that appears at any input to an OR gate will appear at the output. OR gates are very useful, for example, when entering a shift register. If it is desirable to route numbers from several points in a computer into one register an OR gate is used.

AND gates, on the other hand, are used to control the flow of information. Thus, in Figure 12a

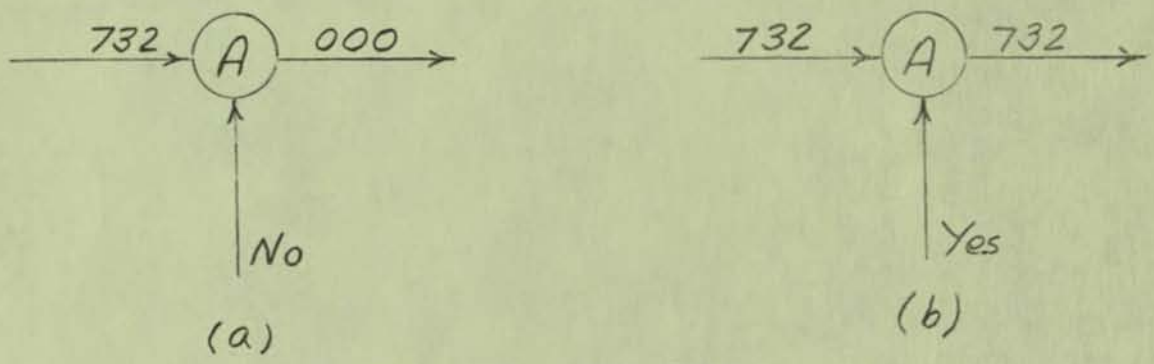
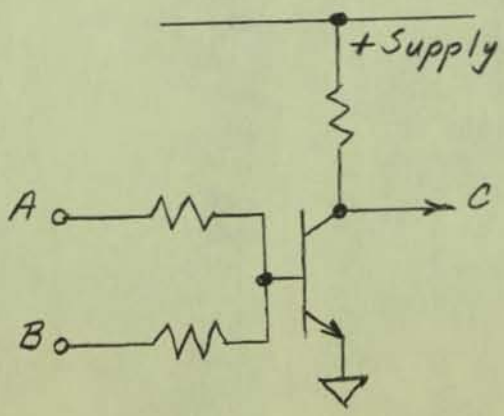


Figure 12. AND Gates

the NO signal appearing at the control input to the AND gate prevents the set of numbers 732 appearing at the other input from appearing at the output. In Figure 12b, where this control signal has been changed to a YES, the set of numbers appearing at the input can appear at the output.

One of the most common gate circuits in use today is the TRL gate shown in Figure 13:



If "1" is a positive voltage and "0" is ground The following "Truth Table" holds

A	B	C
0	0	1
1	0	0
0	1	0
1	1	0

Figure 13. TRL Gate Circuit

This is also commonly known as a NOR gate. Looking at the circuit diagram, we can see that if a positive voltage appears at either input to the circuit, the output terminal at the collector will go to ground, the transistor conducting to saturation. If we call this positive voltage a "1", and the ground voltage a "0", we can write what is known as a truth table which describes the operation of this circuit. We can see in this truth table that if a positive voltage or a "1" appears at either or both inputs, the output terminal C goes to ground or "0". We can also see that if we have a ground potential or "0" at both of the input terminals, the output voltage goes positive to a "1". We say that the "0" condition is the inverse or the NOT of the "1" condition; and hence, we describe this gate by saying that the output is the NOT of the OR of the inputs - hence the name NOR. To get the true of the OR of the inputs, we would have to invert the output of this logic circuit which would be accomplished by going through another transistor. If we put NOT A at one input and NOT B at the other input, that is, if A and B are "1", both inputs to the gate are "0" we can see that the output terminal will be a "1". Thus, we can also say that the output is the AND of the NOTS of the inputs. It is a "1" when A is "0" and B is "0". From the above, it can be seen that either the AND function or the OR function can be accomplished using this same gate structure. In fact, examining all types of logic circuitry it can be shown that all logic can be constucted using this basic logic circuit interconnection in various ways. This is the basis of the modules put out by some companies. These modules are just this basic NOR block. I know that the above is a lot to swallow, but if you are willing to take some time, I think it would be well worth going over again until you understand it.

Of course a micrologic gate circuit is simply a G element as shown in Figure 14:

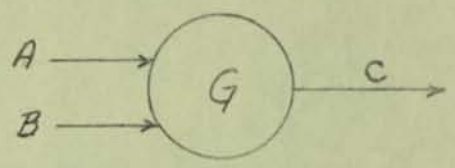


Figure 14. Micrologic Gate Circuit

3. Adder

I will make no attempt, at this time, to go into a full logical discussion of an Adder. Basically, an Adder performs with gates the same operations that you performed in adding two binary numbers earlier. A block diagram of one of the most common adder configurations is shown in Figure 15:

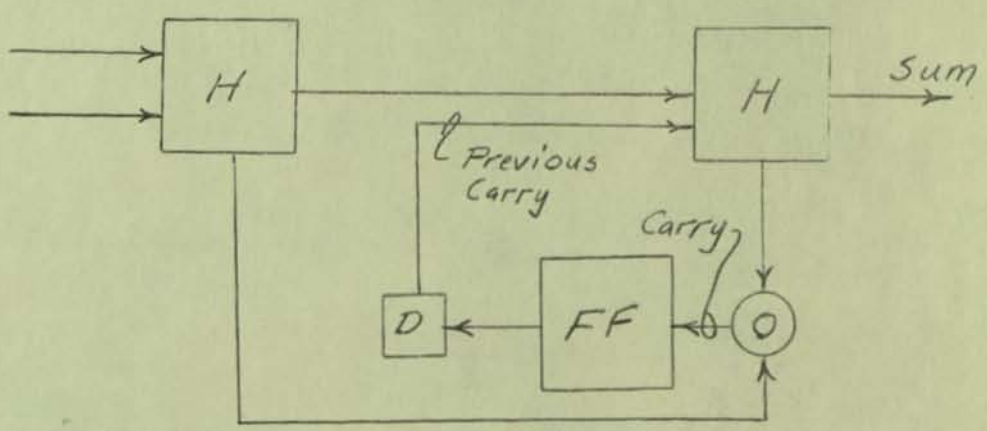


Figure 15. Block Diagram-Adder

The two inputs to the adder are shown as X and Y. The two boxes labelled H are really each a set of three of the TRL

gate circuits shown in Figure 13. These three circuits are combined in a commonly appearing form called a Half-Adder. A Half-Adder provides two outputs; one the sum and the other the carry. The sum output of a half-adder is a "1" when either input, but not both, is a "1". The carry output is a "1" when both inputs are a "1". If you go back to the addition process, you will see the reason for the carry. Now a carry generated in getting the sum of two binary digits must be added to the next higher order binary digits. This is accomplished through the one-bit delay using the flip-flop and delay line as shown in Figure 15. The fabrication of this adder using TRL would require eleven transistors, thirty-four resistors, and two capacitors; and these components would require one-hundred and five printed circuit board holes and interconnections. A micrologic adder is shown in Figure 16:

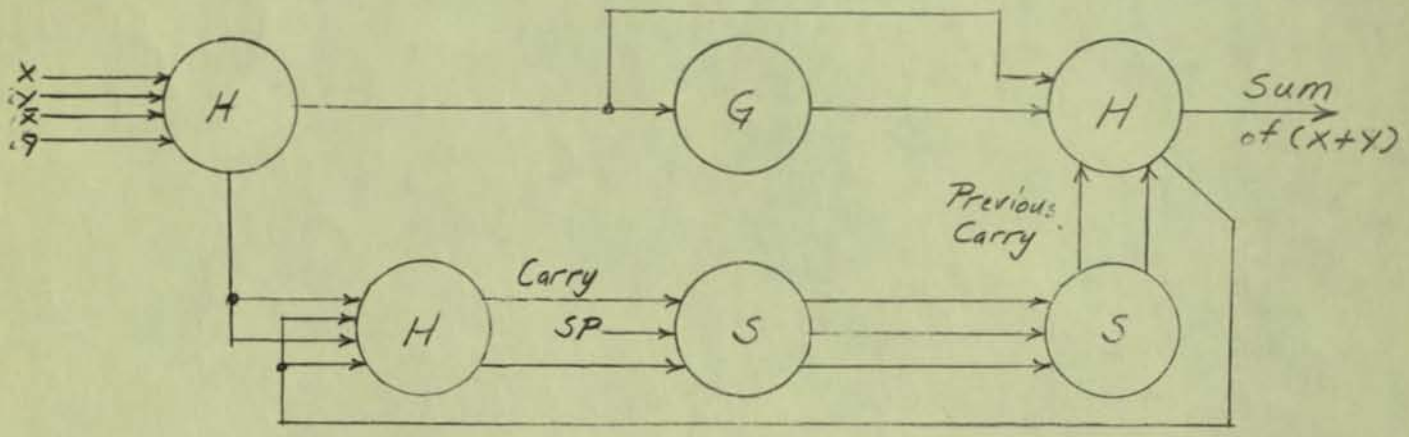


Figure 16. Micrologic Adder

This adder requires 6 micrologic elements and 44 interconnections.

4. Special Cases

Matrices and counters are two logic circuit configurations which are consistent enough in their appearance in all types of computers to be uniquely identified. Their incidence of use in any one computer is very small. Some few computers do not use counters, but nearly all use matrices and counters.

A matrix is a set of gates used for converting from one number system to another. It most often appears in the conversion of binary numbers to decimal numbers. Thus, in the order matrix shown in Figures 1, 2, 3 and 4, a binary order code is inserted into the order register. The matrix takes the ones in zeroes which appear at its inputs in parallel, and provides a "1" on only one of its output lines. The maximum number of output lines for a matrix is 2^n where n is the number of binary digits which appear at its input. In a sense, a matrix performs the same operation that we did in determining the decimal equivalent of a binary number earlier.

The use of counters is probably familiar to all of you. They are used in the same fashion in computers. Pulses are applied at the input stage of a counter chain and the counter counts the number of input pulses. It increments by one digit for each input pulse that appears. A block diagram of a counter stage is shown in Figure 17:

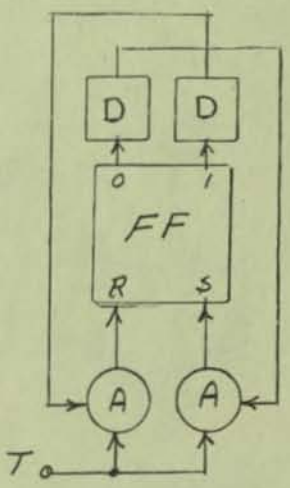


Figure 17.
Block Diagram-Counter

Here we have a flip-flop. The delayed outputs of this flip-flop, feed AND gates. You will see that if the flip-flop is initially in the "1" state or "1" condition, then the AND gate on the right is open and on the left is closed. Thus, when the next trigger pulse comes in, the trigger passes through the AND gate on the right to reset the flip-flop to the "0" state. This causes the action to reverse, etc. The delay lines are required in some form to maintain the previous state of the flip-flop at the inputs to the gates during the trigger pulse. A TRL counter

is shown in Figure 18:

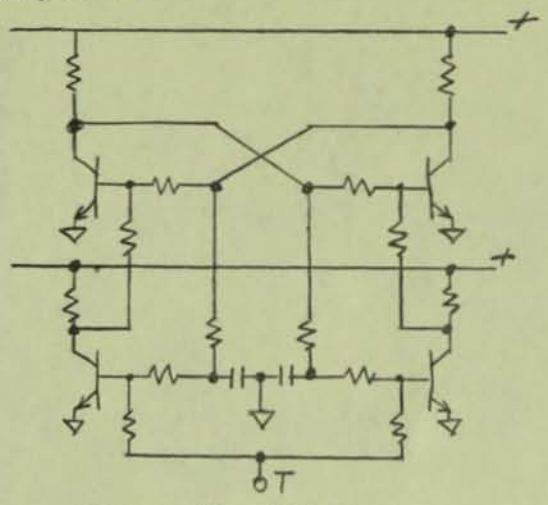


Figure 18. TRL Counter

A micrologic counter stage is shown in Figure 19:

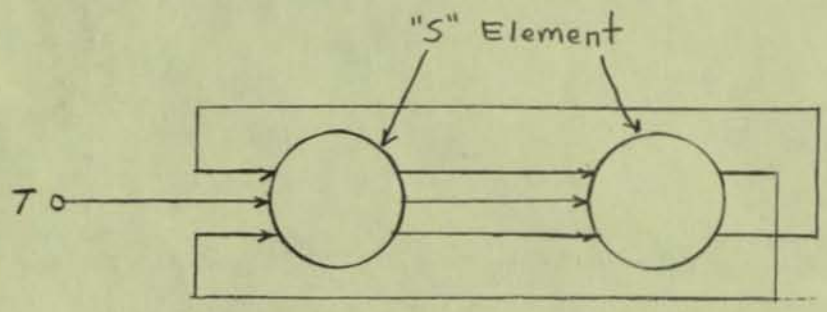


Figure 19. Micrologic Counter

We have examined in some detail the nature of typical logic configurations encountered in a computer or similar data processor. It would now be appropriate to go into the nature of the basic logic circuitry and the reasons for a choice among different types of logic circuitry.

F. Logic Circuitry

Years ago, it was common practice to design what was felt to be a reliable flip-flop, and then design gate structures, and then try to design some sort of circuit to permit the flip-flop to drive the gates. It became apparent, after some time, that it would be easier to design things if the gates were made to look like

one-half of a flip-flop, and the half flip-flop was made to drive so many structures similar to itself, thus, the other half flip-flop becomes just one of those structures and the gates others, and consequently we could begin to talk about a load-driving capability of the flip-flop. This approach was exemplified in what is now the almost universally used technique called distributed logic. Distributed logic is a term applied to a logic circuit concept, which allows that only a few levels of gating will be permitted between, in a sense, buffer amplifiers. In some types of distributed logic, the gating and amplification functions combine. The two common types of this form are TRL and DCTL. Low level logic employs two gating levels between each amplifier section. There are also in use today some variations of the above logic forms, such as R-C Coupled Transistor Logic, and in addition, many forms of what is generally called diode logic. These diode logic forms take the traditional configuration of one or two levels of diode gates followed by an amplifier. These commonly used logic forms will be discussed individually below, along with some notes on advantages and disadvantages of each:

1. TRL

Perhaps the most popular logic circuitry today is TRL. The majority of transistorized digital computers in operation today use TRL. TRL is a very attractive type of logic circuitry. It uses fewer semiconductor components than any other type of logic circuitry, with one exception, it uses fewer components than any other type of logic circuitry. The power consumption is reasonably low. It can tolerate a wide variation in transistor, resistor and power supply parameters without error. Its speed is reasonably high. However, the speed of TRL is only one-third that of either low level logic or DCTL. The minimum power required in TRL is about five times that required in either of the logic forms mentioned above. Nevertheless, the speed which can be achieved with commercially available transistors, and the low power which can be achieved with this circuitry is sufficient for most applications.

2. DCTL

DCTL uses fewer components than any other logic form. It requires less power than any other logic form. It will operate as fast as any saturating logic circuitry, and will tolerate as wide or wider variations in parameters than any other logic circuitry. The catch in DCTL is that a suitable transistor for DCTL must be used in this circuitry. A DCTL transistor must have a high input resistance in saturation. It, of course, as with most other saturating logic must have a low saturation voltage. Historically, only a few transistor types have been suitable for DCTL.

3. LLL

Low-level logic, which is an extension of an earlier logic form, has many of the advantages of both the above described logic circuits. It will operate as fast as DCTL. It requires little more power than DCTL. It retains the advantage of being able to operate over a wide range of parameter variations. It does use more components and more semiconductor components than both of the above logic forms. Some of the comments which apply to the DCTL transistor also apply to transistors which are to be used with low-level logic. However, the nature of low-level logic is such that these requirements are nowhere near a stringent as they are for DCTL.

4. Classical Distributed Logic

Most of the comments which apply to a various clasical distributed logic forms, which is a more general term than diode logic, are on the debit side. This logic form requires orders of magnitude more power than that required for the logic circuits discussed above. Component tolerances are very tight. The use of one percent resistors, for example, in this logic circuitry is common practice. Requirements on the transistors in terms of parameter variation are very stringent in this logic form. Furthermore, the higher voltages required, the higher transistor dissipations which are encountered, etc. put much more stress on the transistor than is encountered in the previous circuits.

Transistors are almost always operated in the non-saturating mode in this logic circuitry, in an attempt to regain some of the speed lost through high voltage swings, high resistances, etc. With few exceptions, the speed of this circuitry is not as high as that which can be achieved with DCTL or LLL. The number components required in Classical Distributed Logic far exceeds the numbers required for any of the logic forms mentioned above. This type of circuitry was only justifiable when transistors with lifetime control and consequent short storage times were not available.

5. Current Steering Logic

The type of logic circuitry which has been used in attempt to achieve high operating speeds is Current Steering Logic. Basically, Current Steering Logic gets around most of the speed problems in any logic circuitry. The voltage swings are kept low to reduce capacitive affects. The transistors are not allowed to saturate to eliminate storage effects, etc. A basic current steering logic module requires several current sources, and consequently, dissipates a great deal of power. An unfortunate characteristic of current steering logic is that complementary transistor types should be used to achieve reliable operation. When only one transistor type is available, so called voltage translating blocks must be used, which in themselves are satisfactory in operation; but as a result of their use, the voltage appearing at the input to the next stage can vary widely in average level depending on variations in the various current sources required in the preceeding stage. Once again, with the controlled life-time transistors, which are now available, and their good saturated switching characteristics, the gain in speed of current steering logic as opposed to DCTL or LLL is marginal. On the credit side, it can be said the current steering logic is probably as insensitive to transistor parameter variation as any

of the other forms mentioned. It is also a logic form well-suited to use with high saturation voltage transistors. Some of the high-frequency transistors which have been available in the past, have had this unfortunate characteristic.

The above summarizes some of the thoughts on the various logic circuit configurations. The circuitry chosen for micrologic was DCTL. The only basic reason for preferring this circuitry to low-level logic was the significantly reduced number of components. DCTL, when used from the standpoint of simplicity of construction was an ideal choice for micrologic. It was felt, at the time that the decision was made to go to DCTL, that a suitable DCTL transistor could be developed. This has been the case. In view of the unsuccessful efforts by some of the biggest silicon transistor manufacturers in the country to develop a DCTL transistor, this alone can be considered quite an accomplishment.

G. Why Micrologic

The comparisons made above of micrologic configurations with TRL configurations for some typical logic circuits, point out some of the chief features of micrologic in computer construction. The number of components which must be handled is significantly reduced. This effects two savings to the computer manufacturer. In the first place, the number of components which must be tested, handled and assembled is significantly reduced. Secondly, the loss in reliability due to the testing, handling and interconnection of these many components is significantly reduced. The simplicity of design as shown in the illustrations above result in reduced engineering time to the design of a computer, and much reduced design time to Class A drawings. All these little things save the equipment manufacturer money. We will go into some of the other characteristics and features of micrologic in the next section.

III. Micrologic

You have all read, I am sure, some of the various status reports and perhaps even paper reprints on micrologic. I hope that this familiarization program has helped you to understand better the significance of

what was said. We have said it so many times and in so many ways that it tends to sound trite to us, but sounding trite should not detract from its true significance to the equipment manufacturer. What is micrologic? Micrologic is a name applied to our approach to computer miniaturization. We use the term micrologic because we wanted to do something different. We said that the only people who would be interested in how many of anything you could put in a cubic foot or how many of those things you could put in a thimble or how many of them would fit in gnat's eye are the public relations people.

The computer manufacturer is basically interested in how much computer he can pack in a cubic foot,or saying it another way how many cubic feet or cubic inches he can reduce the size of a given computer down to. This carries some implications. It says that a miniaturized part to be used in constructing a computer should be capable not only in itself of being small but it should be capable of easy assembly into a subminiature machine.

We have mocked up for the Eastern Joint Computer Conference the logic section of a 22-bit digital computer. This includes the control and arithmetic units using the TO-5 size micrologic elements on printed circuit boards. This logic section would be approximately 6 x 12 x 1 inches. Using the TO-18 size micrologic elements with welded wire techniques, this logic section would be 6 x 1-1/2 x 3/4 inches. Keep in mind that this is the full logic section. The only other parts to the computer are the storage,either a core or a drum memory, and the power supply. This is the kind of information that a computer manufacturer is interested in knowing.

Micrologic elements were designed for interconnection. They were designed this way so that in the first place they could be interconnected using contemporary techniques and secondly so that they could be interconnected without wasting a lot of volume running leads back and forth. They were further designed to dissipate low enough power so that it would not be necessary to waste a lot of volume with heat radiators. In short, they were designed not just to make the pieces small but to make the whole assembly small.

This attitude has prevailed throughout our approach to various

considerations in micrologic program. The selections of the various functional blocks to be used in micrologic were based on end use considerations. The decisions as to what type of fan-in and fan-out to design the micrologic for were based on end use considerations. Micrologic is designed to be applied. It would be an extremely expensive novelty to put this much money in for something we can point to and say, "look fellows, it's small". It's designed for real computers which we expect to see flying in two years.

We say then that we have an approach that is designed to be used. What do we have to offer to the computer manufacturer that should encourage him to use it? We have a compatible set of digital functional blocks each fabricated on a single chip of silicon and now in a hermetically sealed package. A complete logic section can be fabricated using generally speaking only micrologic elements. This logic section will be capable of operating over the full military environmental range including a temperature of -55°C to +125°C. This logic section will be capable of operating at speeds as high as 1 mc.

The cost of this logic section when fabricated will be less than 1/4 that of a logic section fabricated using transistors and resistors. This logic section will probably use less power than one fabricated using transistors and resistors. It can be one to two orders of magnitude smaller in size than one fabricated using transistors and resistors, depending on the type of packaging used. It will be at least as reliable and probably much more reliable than one fabricated using present techniques. It will be easier to design and easier to manufacture. This then is what we have to offer. This is micrologic. The cost savings and design and manufacture are so great that even at the selling price of the sample quantities of micrologic elements. The cost of a logic section using these elements will be no greater than that of one using transistors and resistors.

Some people will be interested in micrologic because it represents to them an easy way to break into the digital computer field. Those who are already in the field should be interested because micrologic elements represent a reliable useful, economical way to build a digital computer. A lot of what I have said, and much more of course, appears in various

reprints and status reports, etc. If you don't already have copies of all this information, please make sure that I get it to you. The logical question that comes up next is how does micrologic compare with the other miniaturization approaches which are currently flooding the field. This will be discussed in the next section.

IV. Comparison of Micrologic to other Approaches

A. What is the best Miniaturization Approach to use

It would seem that the heading of this section is one of the most often asked questions in the electronics industry today. There is no pat answer. The best miniaturization approach to use depends heavily on what is to be miniaturized. Those cases where integrated circuits, which are available in large quantities, can be used should use these blocks. When this is not the case, due to the special nature of the circuitry, or due to the use of circuitry which does not readily lend itself to the batch processing features of integrated circuit construction, should use a more flexible miniaturization approach. Another important consideration in the choice of miniaturization approach is the time factor. Approaches which are well suited to two or three year delivery time of prototype equipment may be completely out of the question for a one or one and one-half year delivery time. This section will generally discuss all microminiaturization approaches. Some examples and the features and disadvantages of each approach are also given. Before proceeding, it might be worthwhile to discuss some of our own opinions on this subject.

B. Opinions

The comments contained in this section of the familiarization program represent our own "objective" appraisal of the various approaches. Certainly to some extent this appraisal would be subjective. But to clarify our position, it would be worthwhile to sum up our own opinions as to which approach is the most suitable. The one microminiaturization approach which will yield the fastest time to delivery of a prototype of anything is known as the "Cordwood Approach". This is represented by the various welded stick and welded module approaches to miniaturization. Some of these are the Sippican Francis Associates, EECO, The Raytheon Weld-Pack, Delco, Sprague Welded Modules, and others. This approach is capable of about a five-to-one reduction in equipment size. This is a

significant volume reduction. It has the advantage that it permits the user to assemble his own reliable circuitry using components in whose reliability he has already established confidence. This is the only miniaturization approach for which this is true. The welded wire interconnecting technique seems to be a reliable one when properly implemented.

As soon as we depart from the above approach, we run the average time to delivery on the prototype out to about two years. Any other approach requires an evaluation of the resulting functional blocks or circuits or what have you in miniaturized form to determine their reliability. This involves a great deal of time. Coupling this time figure with the time required to design in the new approach, and the time perhaps to availability of suitable quantities or suitable characteristics in the new approach, set the two-year date. In the area of run-of-the-mill circuitry, particularly logic circuitry, integrated circuits represented by Fairchild Micrologic, TI Solid Circuits, and later on the Sperry Seminets are the best choice. These approaches will yield the smallest size of prototype equipment coupled with lowest potential cost, and I feel, highest reliability of all available approaches.

Circuitry which cannot be considered run-of-the-mill can become very expensive, even prohibitively expensive, to assemble using the integrated circuit approach. Thus, even at the two-year level, the Cordwood approach represents, I feel, the best choice. A logical extension of this Cordwood technique is the approach taken by our own Special Products Division. Rather than package transistors in individual cans and then assembling circuits multiple transistors and even circuits are packaged using transistor production techniques in one can. The cost of this approach is not very high. Other savings may be realized in the Cordwood approach by using the time available to evaluate the reliability of subminiature components which are well suited to this technique. Another type of microminiature approach better suited to the non-standard circuitry is the Micromodule approach which deals with individual circuit components on wafers, and thus allows design engineers considerable flexibility in the design of circuitry. The Micromodule approach represents the first, albeit long, step below the Cordwood approach in circuit flexibility.

This concludes the opinions section. Now to go on to opinions.

C. Microminiaturization Approaches

Some examples and characteristics of the more popular approaches to microminiaturization will be discussed below. The techniques which will be discussed are listed as follows:

1. Subminiature packaging of standard components.
   a. Subminiature packaging of non-standard components.
2. Subminiature packaging techniques using deposited passive components and standard or near standard active components.
3. Integrated circuits
4. Functional Blocks

It is worth noting at this point that only the first type of microminiaturization approach does not involve an extensive reliability evaluation of new components and/or circuits. It is further worth noting that all of the techniques described above probably will be used. Poor sales of an individual manufacturers' microminiaturized parts will generally not be due to a lack of a market for his approach, but more likely due to the failure of performance of his parts.

1. Subminiature packaging of standard components.

Examples of this technique of microminiaturization are the so called Cordwood approach and variations, thereof, and the Burroughs Macromodule approach. This approach as mentioned earlier yields at least a five-to-one reduction in size. Examples of the Cordwood approach are the Raytheon Weld-Pack, some recent additions to the EECO product line, the Marion Associates Sippican computer, and others. There seems to have been a considerable acceleration in the acceptance of this technique as a useful stop gap. Modules using the welded wire interconnection technique are being made by such companies as Delco, Sprague, and of course, EECO as mentioned above. This approach basically involves more efficient packaging techniques for standard components. The significant feature of this approach, as stated before, is that a manufacturer's own reliable circuitry can be packaged efficiently using this technique. The components which are used, since they are standard parts, are easily available. These components can be easily tested before assembly into modules. Because of the standard nature of these components,

their reliability is already known to the equipment manufacturer. The basic disadvantages to this approach involve the welded wire interconnection technique. The cost in the amount of skilled labor required to get reliable interconnections in large assemblies is significant. Maintainability of this type of module is practically impossible. Modules fabricated using this technique would have to be considered as throw-away parts.

a. Subminiature packaging of non-standard components.

Some examples of this approach to microminiaturization are the Micromodule of RCA, the Hughes Dot System, and the Mallory approach. This approach involves the fabrication of individual components in small compatible packages. These packages are then assembled to construct circuits. The chief advantage of this approach is that some of the implementations will yield a quite small size for circuitry which would be otherwise difficutl to microminiaturise. Size reduction of the order of 5 to 10 to 1 can be achieved. Once again, with this technique, it is possible to test individual components. Some of the disadvantages of this approach involve the low-volume efficiency of circuits microminiaturized this way. In addition, the availability of the components in question is in doubt at the present time. No doubt many of the components used in this approach will be available perhaps within the next year.

2. Subminiature package containing deposited passive components and "standard" active devices.

Some of those quite active in this field of microminiaturization are Sylvania, IBM, Arma, Diamond Ordnance Fuse Lab., and others. Some of the components which are used on these substrates can be tested before assembly. One of the basic disadvantages of this approach is that it must still be considered to be in the developmental stage. Consequently, some time must pass before useful microminiaturized components of this type are available. Furthermore, assembly costs on the basic microminiaturized module can be quite high as well as assembly costs of these modules into systems. Some component limitations exist with this approach, and the incentive to expand this technique is doubtful.

3. Integrated Circuits

Fairchild Semiconductor, Texas Instruments and Sperry Semiconductor, among others, are actively involved in the development integrated circuits. The use of integrated circuits can result in equipment volume reductions of one to two orders of magnitude. One of the chief potential advantages of this approach is its very low cost compared with any other. The techniques used in fabricating these integrated circuits are for the most part the same as those techniques used in the fabrication of transistors. Work by various people in the area of reliability evaluation of integrated circuits indicates that they will be highly reliable. Of course, it must be kept in mind that the computer manufacturer will have to establish his own confidence in the reliability of integrated circuits as well as any other technique. These advantages, however, do come out of the batch processing techniques characteristic of some transistor manufacturing methods. This technique does have a couple of disadvantages. One of these involves the restricted number of functions which can be available in the integrated circuit form. Another disadvantage which could be significant involves the inability to test the components within the integrated circuit.

4. Functional Blocks

This category is basically included for the sake of completeness. Some great things can be done in the future in the area of functional blocks. Keep in mind the functional blocks are not integrated circuits. They are chunks of material that perform a function such that individual areas or regions cannot be identified as an analogue of a specific component in the circuit which is replaced. Functional block work is still pretty much in the research stage.

D. Advantages of Micrologic Compared to other Integrated Circuit Approaches

The first comment I have to make with respect to the title of this section is caution in a profit making system. Any advantages one approach has to another of the same approach type, will be as temporary as your competition can possible make it. If feel that some of the basic relative

features of micrologic to other approaches are its high performance, its low cost both initially and ultimately, and its usefulness to the equipment manufacturer. Usefulness, of course, as we have discussed before involves not only the performance characteristics of the integrated circuit, but also, the manner in which it lends itself to accepted packaging techniques.

V. Applications Support

Now that we are finally drawing to the end of this familiarization program, a few minutes will be spent discussing the type of applications support you can expect for your sales effort in the field. In general, this support will be similar to the support you receive in your transistor sales effort.

To start things off, we have, at the present time, a list of names of people who have been interested enough in micrologic to write to us and ask to be put on the mailing list. This involves hundreds of people. You may have been bothered in the past by our insistence on keeping this list, but this was its purpose. As part of this list, we have a much smaller list of those people who specifically requested design information on micrologic. One might say that their interest is perhaps more intense. In any event, this list will be available to you to serve as contacts.

Various papers and many "status reports" have been written on the subject of micrologic. Other papers are in preparation. The micrologic applications handbook has been prepared for those who are specifically interested in design information. Reprints of all papers and other publications will henceforth be available to you for distribution at your discretion to interested parties. It will be in your own interest to supply us with the names of those who receive this information so that we may build up a second micrologic mailing list which will assure that your customers will receive all subsequent published information.

We will be available for direct contact at your invitation with your customers. I imagine that this will be desirable, certainly in the early selling stages of micrologic. We do, however, prefer to postpone this direct applications support effort until saleable units are made available.

If you would rather have us visit someone before that time, please let us know.

Of course, we will give you all the home office applications support we possibly can. We have done a great deal of work with the micrologic elements in putting them together in typical logic configurations; looking at the engineering problems, design problems, packaging problems; and all this information is available to you at your request. Well, I guess all we can say now is lots of luck. Thank you.